Integration of SATURN System and VGIS*

Hu Yu
Department of Computer Science
University of Illinois at Urbana-Champaign

Sharad Mehrotra
Department of Information and Computer Science
University of California at Irvine

Robert Winkler
Sean S. Ho
Timothy C. Gregory
Swati D. Allen
Army Research Laboratory

ABSTRACT

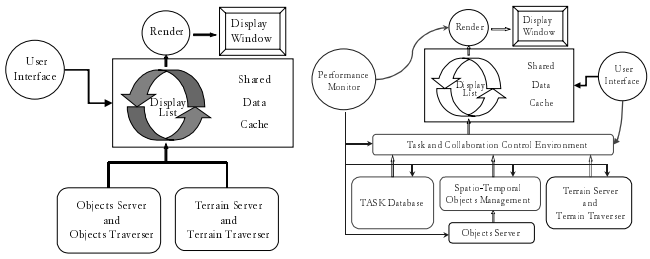
This paper reports on the integration of the SpAtio-Temporal Uncertainty ReasoNing (SATURN) system being developed in our group with the Virtual GIS (VGIS) system in order to improve its performance and scalability to complex dynamic environments as well as to enhance its functionality as a collaborative planning tool. To achieve this we added three new components to VGIS: a spatio-temporal object manager, a performance monitor, and a task database. The spatio-temporal object manager uses SATURN techniques for indexing dynamic multidimensional (spatio-temporal) objects to support effective and efficient object traversal during visualization. The performance monitor adjusts the resource allocation between VGIS components and adaptively adjusts image quality to guarantee bounded visualization performance. The task database extends VGIS as a tool for collaborative planning. Performance results illustrate that the SATURN techniques for object management and the performance monitor significantly improve VGIS performance allowing it to scale to complex scenarios with a large number of dynamic objects.

INTRODUCTION

This paper reports on the progress we made in the integration of the SpAtio-Temporal Uncertainty ReasoNing (SATURN) system with the Virtual GIS (VGIS) system in order to improve the performance of VGIS during interactive visualization in dynamic environments as well as to extend its functionality as a collaborative planning tool.

SATURN: SATURN is an ongoing project in our group whose goal is to develop and empirically validate efficient techniques to manage consistency and uncertainty in dynamic, rapidly changing multidimensional (spatio-temporal) databases in order to support effective visualization and intelligent information processing tasks. The SATURN technology [11,12,13,14] includes effective techniques for indexing multidimensional data sets, techniques to integrate multidimensional data structures into DBMSs as access methods, techniques to support concurrent operations on multidimensional data structures, representation and management of dynamic spatio-temporal objects (e.g., moving objects in a battlefield), support for uncertain conceptual spatio-temporal queries, techniques for query refinement in databases as well as query processing mechanisms that optimize and effectively handle uncertain queries.

VGIS: Virtual Geographical Information System (VGIS) is a 3D terrain visualization system that supports visualization of global multi-resolution terrain elevation and imagery data, static and dynamic 3D objects with multiple levels of detail, and distributed simulation and real-time sensor input developed at ARL. Figure 1(a) shows the VGIS architecture.



(a) before integration (b) after integration

Figure 1. VGIS Architecture

* Prepared through collaborative participation in the Advanced Displays and Interactive Displays Consortium sponsored by the U.S. Army Research Laboratory under Cooperative Agreement DAAL01-96-2-0003.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
1999		**00-00-1999 to 00-00-1999**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Integration of SATURN System and VGIS	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of Illinois at Urbana-Champaign,Department of Computer Science,201 N. Goodwin Avenue,Urbana,IL,61802-2302**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**6**	

In VGIS the server/traverser threads communicate with the external processes and fetch data from the local disk, as well as translate the data format. For each user (view), there is a terrain server/traverser that maintains and executes data requests to the terrain data set and handles level of detail management, terrain rendering, etc. Similarly, there is an object server/traverser that manages storage, animation and display of 3D symbology and protrusive objects. The scenes that are to be rendered are prepared by the server/traverser pairs and buffered in data structures similar to OpenGL display lists. The actual on-screen rendering is done by a single render thread, which parses the most recently updated associated display list, and displays it on the display window.

Existing VGIS technology suffers from two performance limiting issues. First, the object server/traverser pushes graphical information of all objects that were involved in VGIS to the display list, when a new frame is generated whether or not the object is outside the user's field of view. This severely limits the performance of the existing VGIS during interactive visualizations in environments where large number of dynamic objects are involved. The second issue is that all threads in VGIS run in a "best effort" fashion. Although the disadvantage is not significant when the system resources are adequate, the "best effort" model does not guarantee bounded performance (e.g., guaranteed frame rate) in case of large job sizes and limited resources.

Integration of SATURN with VGIS: In order to overcome the above two performance related impediments of existing VGIS as well as to extend its functionality as a collaborative planning tool, we explored the integration of VGIS with SATURN. The integrated system is illustrated in Figure 1(b). In the integrated system the following three modules have been added to VGIS.

1. *Spatio-Temporal Objects Management:* Appropriate representations of dynamic objects, indexing mechanisms and support for concurrency provided by SATURN were applied to provide efficient and effective support for managing objects in motion as well as static objects.
2. *Performance Monitor:* Uses a performance-driven resource allocation approach to make appropriate tradeoffs among different VGIS threads to adaptively adjust the quality of the vizualization. The performance monitor attempts to guarantee bounded performance satisfying the various Quality of Service (QoS) requirements of the visualization.
3. *TASK Database:* The task database, including "Task and Collaboration Control Environment", provides a mechanism to model and store tasks used to drive the virtual world or any interactive visualization in a database. Such a task database enables user(s) to checkpoint the state of the virtual world at any instance, backtrack to an earlier location in space and time, define alternative timelines, navigate freely in both space and time in the virtual world using a powerful spatio-temporal query interface, as well as to store, modify and resume old tasks in different sessions with the visualization system. The task database extends the functionality of the VGIS system as a spatio-temporal collaboration system in order to facilitate its usage in diverse applications such as mission planning, rehearsals, and training.

In the following sections, we describe the spatio-temporal object management and the performance monitor modules we added to VGIS in further details. Due to space restriction, the discussion on the task database extension to VGIS is not included and will be reported elsewhere.

SPATIO-TEMPORAL OBJECTS MANAGEMENT

Motivated by the requirement to handle large number of objects in battle simulations and the limitations of existing VGIS system, we integrated SATURN's spatio-temporal objects management techniques into VGIS. The objective is to provide efficient and effective support for objects in motion, such as moving vehicles, weather objects, NBC objects and buildings. Spatio-temporal objects managed using SATURN are used in VGIS for two purposes:

- The VGIS system, in order to generate the next frame, queries the database for objects that will be within the field of view of the user. The query is run every time VGIS generates a new frame. The resulting objects are dispatched to the display cache to be rendered to the screen. As will become clear, the objects are maintained and retrieved from the database at the resolution at which they will be displayed based on the elevation of the user.
- During a visualization, a user may query the database to retrieve objects that will be seen in the future (e.g., "how many enemy tanks will be within my firing range over the next hour", or objects the user saw in the past (e.g., "was I in the firing range of any enemy vehicle in the past"). Such queries are specially relevant in the context of the task database which stores the history of the user interaction in the context of a task within the 3D virtual world.

Based on the above, we classify the VGIS interaction with the SATURN database into the following queries:

- *Current Query*: the query retrieves objects whose attributes satisfy a specified predicate at a given in-

stance (e.g., retrive objects in my field of view for visualization purposes).

- *Future Query*: the query retrieves objects on a projected future based on the information about objects and their trajectories stored in the database.
- *Past Query*: the query retrieves objects based on the historical information about objects and their spatio-temporal properties stored in the database.

Different representations of spatio-temporal objects are suitable for different types of queries. Specifically, an *Original Space Representation* (OSR) is used to support historical and current queries while *a Velocity Based Representation* (VBR) performs well for future queries.

- *Original Space Representation:* An object is represented as a 3D poly-line based on its X, Y, and time coordinates. The starting point of each line segment is determined by the position of the object when it sends its initial location to the database as well as the time at which the update was sent. The end point of a line segment is determined by the location and time of the next update, or by projecting the object's future position based on it's current velocity and direction. In this representation, a query corresponds to a bounding rectangle in space and time. The answer set consists of all the objects that have line segments that intersect the query rectangle.
- *Velocity Based Representation:* This representation maps an object between two updates to a point in a 5D space where the dimensions correspond to the starting x and y positions, velocity in the x-direction , velocity in the y-direction and time. A trajectory of an object corresponds to a set of points in the 5D space. In this representation, a spatio-temporal range query corresponds to a set of objects represented by points within a parallelogram in the 5D parametric space.

Our experimental results over real data sets illustrate that OSR is suitable for historical and current queries while the VBR representation allows for significantly better performance in processing future queries. For this reason, two copies of motion information of objects are maintained. One represented in OSR to answer historical and current queries and another using VBR to answer future queries. We therefore refer to objects represented in OSR as the *Historical Database*; while *Future Database* is used to refer to the objects represented using VBR. Maintaining two representations imposes an update overhead as well as an additional burden to maintain the consistency between these two copies. Consistency between the two representations is critical when answering a mixed query that refers to both past as well as future information (e.g., retrieve information about all friendly tanks I saw over the past 10 minutes as well as information about tanks I will see in the next hour). Additional overhead of maintaining two consistent representations is however mitigated by the improved performance of such queries.

In order to support efficient retrieval, multidimensional indexing mechanisms of SATURN are used to index both the OSR and the VBR representations of moving objects. Specifically, a modified R*-tree that supports a parallelogram based range query is used for this purpose. Notice that since the data set is dynamic (every update causes the spatio-temporal location of objects to be modified), the R*-tree must support concurrent operations in which insertions / deletions of objects in the index structure execute concurrently with the search queries. Notice that sequentializing this access by blocking searches while an insertion/deletion occurs in the tree will result in very poor performance. SATURN uses a *dynamic granular locking solution* reported in [11] that provides high concurrency with low locking overhead in order to support concurrent operations over multidimensional data structures.

Another key issue in the VGIS and SATURN integration is that of policies used to update the location of the objects in the spatio-temporal database. In a real battlefield, immense amount of spatio-temporal information such as unit and weather object movement and tactical operations are collected from various different sensors, radars, and reports. The location of objects in the database needs to be updated whenever the real object deviates from its representation in the database. An update consists of a objects new location, direction, velocity, and time. Even though we have not explored this in the context of VGIS, advanced strategies may also allow objects to predict their expected future path as well. Many update policies and their effect on the degree of precision about the location of objects in the database have been explored in [10,11]. In VGIS and SATURN integration, we support three different update mechanisms.

1. Updates are sent at evenly spaced time intervals. This guarantees that the object's position can deviate from the stored position by a bounded amount, depending on the length of time between updates.
2. Updates are sent when there is an aspect change in object motion, e.g. the direction or velocity changes by a certain amount.
3. Updates are sent after objects deviate from their projected position by a specified amount, offering the same guarantee as policy 1.

All the above strategies guarantee that the database state deviates from the object's accurate location by a bounded amount. This bound may, however, vary based on the specific policy being used.

Integrating VGIS with SATURN's spatio-temporal data management techniques allows VGIS to scale to domains where the virtual world consists of a very large number of spatio-temporal objects. The primary benefit arises since efficient indexing techniques facilitate the culling of objects which are not within the field of view of a user within the database system. These objects would instead have been sent by the object manager to the display lists and then culled out by the rendering engine in the original VGIS implementation. Our performance results confirm the improvement and scalability gained by such an integration.

PERFORMANCE MONITOR

In the previous section, we described how SATURN techniques allow the render process to fetch 3D symbology and protrusive objects for visualization in a more efficient way. Another component which we implemented into VGIS to improve its performance is the *Performance Monitor* (PM). The idea behind the performance monitor is to adjust resource allocation pattern and image quality adaptively to guarantee and maintain satisfactory performance. The performance monitor considers the quality of the visualization along various *QoS dimensions* that are parameters used to quantify multiple attributes of the overall quality of the visualization. Specifically, we consider the following five QoS dimensions. In the following discussion, we will refer to the number of scenes generated by the server/traverser pairs per second as the *Produce Frame Rate*, and the number of frames displayed by the renderer thread as the *Render Frame Rate*.

1. *Temporal distortion:* (also referred to as the Data-to-Simulation-Delay). This measures the temporal delay between collecting data by the system to displaying the data to the user due to the processing time required by traversers and the renderer to generate and display a new frame. Temporal distortion causes the information visualized by the user to be stale. In order to enable user to percieve accuracy of movement, temporal distortion must be guaranteed to be a bounded value. This goal can be achieved by maintaining produce frame rate higher than a given threshold, which is proportional to objects' maximal velocity and operation speed with respect to the user. The constants can be found experimentally. Note that here we have disregarded the sensor uncertainty , communication delay from sensors to the database and storage uncertainty in database for simplicity. In other words, we assume the data in the database is in exactly the same status as the real world, when VGIS starts to process it.
2. *Content continuity:* Consider two successive frames generated by traversers. A user may feel discontinuity if the two frames differ significantly in their content. To avoid this, we specify an upper bound on the time interval between two successive frames. This also constrains the produce frame rate to be no lower than a threshold, which is proportional to the user's speed, as well as inversely proportional to the user's height. *Render frame rate:* Depends on the view property of user. To gain animation effect, the render frame rate shouldn't be lower than 10-24 frames per second. Again, the constants are found experimentally.
3. *Image quality:* Depends on the resolution used for terrain data and object data. The higher the resolution, the better the image quality.
4. *Frame size:* larger frame size provides better visualization effect.

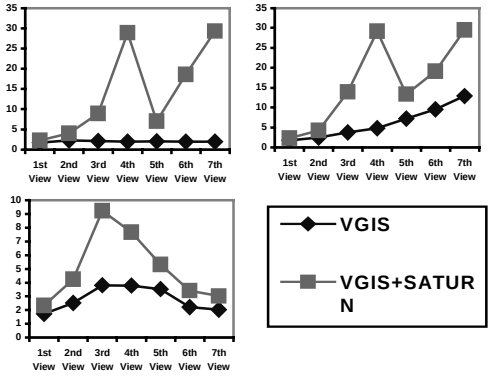
The performance monitor attempts to maintain the quality of the visualization based on the above discussed QoS dimensions. For each QoS dimension, the monitor first determines various constraints as well as feasible regions. During an ongoing visualization, if a QoS parameter drops significantly, or even worse, if a constraint is violated, the monitor attempts to rectify the situation by trading another QoS parameter whose drop in quality will not cause the overall quality of the visualization to drop to an infeasible region. The primary mechanisms used by the monitor to adjust the quality of the visualization are to modify the resource allocation to the different VGIS threads and to control the resolution level for data. For example, if the monitor is triggered by a sudden drop in the render frame rate, it will first determine an optimal resource allocation strategy within the various VGIS components so that threads that are starved for resources (causing the frame rate to drop) are allocated more resources. Besides resource reallocation, depending upon the current display conditions and constraints, the monitor adjusts the resolution levels used in the server/traverser to reduce the job size of relevant threads. These two mechanisms together attempt to maintain the overall quality at a stable and appropriate level without violating any constraints. In our current implementation, , we only consider CPU allocation and the resolution level for spatio-temporal objects stored in SATURN to control the quality of the visualization. In the future we may include other resources, such as memory, into our model, as well as integrate into the monitor a mechanism to control the multi-resolution level for terrain.

EXPERIMENT AND PERFORMANCE ANALYSIS

To test whether integration with SATURN produces more appropriate quality than the original VGIS with neither

spatio-temporal index structure nor performance monitor, we ran a set of tests with a static birds eye view. Experiment was conducted on an SGI RE2 with four 194 MHZ IP25 processors, an R10000 main CPU and an R10010 FPU with 1024 Mb of main memory. For each version of VGIS, three sets of statistics were collected. First, VGIS was run with the mesoscale weather data and the AHAS data. The weather data consisted of 3888 wind vectors, each of that was represented as a 3D arrow consisting of 8 polygons. Then the AHAS data with 425 vehicle object models in its initial state with an average of 100 polygons each was added to the weather data. Second, Replicated AHAS data to 4470 objects with an average of 100 polygons each. That with the 3888 from weather results in 8358 objects in VGIS. Finally, we replicated AHAS data to 44700 objects with 100 polygons each. That with the 3888 weather objects results in 48588 objects in VGIS.

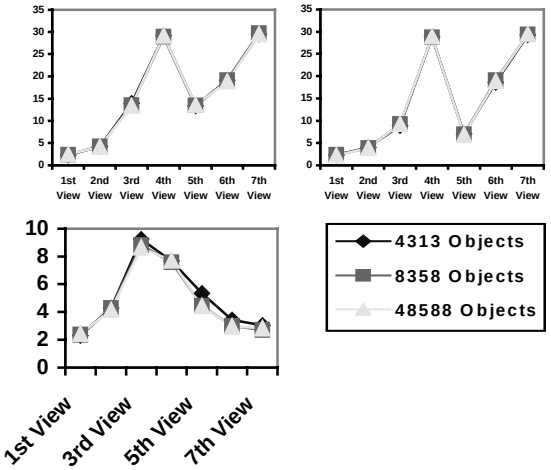
For each phase, statistics were gathered over a period of 60 seconds using 7 different static views: The first four with large height and large view frustrum, one at 8328341 meters height where all of the weather objects were in the view, a second at 4326175 meters height where approximately 1/2 of the objects were in view, a third at 2014524 meters height where approximately 1/10 of the weather objects were in view and a fourth at 63038 meters height where none of the weather objects were in view. The last three with lower height and smaller view frustrum, a fifth at 15935 meters where approximately 100 vehicles in view, a sixth at 2230 meters with 56 vehicles in view and a seventh at 1023 meters with only 2 tanks in view. We collected three statistics for each run of the experiment: Render Frame Rate, Object Produce Frame Rate and Terrain Produce Frame Rate. Figure 4 shows plots of these three frame rates (frames per second) for each viewpoint using the first dataset: weather objects and 425 AHAS objects.



(a) Render Frame Rate (b) Object Produce Frame Rate
(c) Terrain Produce Frame Rate

Figure 4. Plots of frame rates with the first dataset

From the results on the first data set, we are able to say that the performance improvement for VGIS is significant after integrating with SATURN. The render frame rate of VGIS+SATURN is 1.3~6.1 times to that of VGIS; the terrain traverser produce rate increase to 1.3~2.4 times; while the object traverser produce rate of the integrating version is 1.3~14.5 times more than that of the original version of VGIS. Also we can see that, the spatio-temporal index structure and performance monitor work better in cases when less percentage of objects and less complexity of objects' graphical information (i.e., polygons) appeared in the field of view, since the render frame rate and the object produce frame rate keep rising from the 1st to 4th viewpoint, and from the 5th to 7th viewpoint. Note that there is a sudden valley at the 5th viewpoint, this is because the 4th viewpoint contained no objects in the field of view, while the 5th viewpoint contained approximately 100 complex vehicles in view. Another important aspect is that even though in most cases the performance monitor can guarantee and maintain satisfactory frame rates, the first two viewpoints provide render frame rates lower than the threshold given above. Further experiments show that the render thread is not able to display large amount of moving objects and produce satisfactory performance at the same time (There are 3800 and 1900 objects involved in the field of view, respectively, for the first two viewpoints), even if we suspended all other relative threads and assigned the lowest resolution level for each object. To avoid this, only adjusting the resource allocation pattern and the resolution level for objects is not enough. More efficient graphical representation and interpreting mechanism should be applied in the render thread.



(a) Render Frame Rate (b) Object Produce Frame Rate
(c) Terrain Produce Frame Rate

Figure 5. Plots of frame rates for three datasets using VGIS+SATURN

After replicating AHAS vehicles, we constructed the next two datasets , one with 8358 objects and another with 48588 objects, to test the scalability of our system in regard to the size and complexity of object data set involved. According to our experiment, VGIS could not handle such large data sets before integration, and would run out of memory. Figure 5 shows plots of three frame rates (frames per second) for each viewpoint in three datasets using VGIS+SATURN. To make the comparison more accurate, we ensure that the same group of objects appear in the field of view for the three different data sets. I.e. replicated objects only located outside the user's view frustrum.

As we can see, SATURN's indexing techniques scale VGIS to very large datasets. In case that the size of moving object dataset have very little influence to the frame rates, as long as the size and complexity of objects located in the field of view keep unchanging. This property makes it possible to simulate complicated real battle in VGIS.

CONCLUSIONS AND FUTURE WORK

This paper describes our work of integrating SATURN system with VGIS, and implementing spatio-temporal object management, performance monitor and task database to improve the overall performance, functionality and scalability of VGIS. Our test show that the integrated version generates much better performance than previously, thus making it possible to simulate complicated realistic battle scenarios in VGIS. Furthermore, even though due to space restrictions we did not describe it in this paper, the task database enhances the functionality of VGIS as a tool for collaborative planning. We are currently continuing our research on processing uncertain conceptual queries in SATURN. In the future, we will explore integration of SATURN's support for uncertain queries with VGIS to facilitate VGIS as a tool for situational awareness. We will also explore more extensible and scalable models for managing quality of service in the performance monitor.

REFERENCES

1. David Hull, Arjun Shankar, Klara Nahratedt, and Jane W.S. Liu. An end-to-end qos model and management architecture. 1998.
2. David Koller, Peter Lindstrom, William Ribarsky, Larry Hodges, Nick Faust, and Gregory Turner. Virtual gis: A real-time 3d geographic information system. In Proceedings Visualization '95, pages 94--100.
3. Peter Lindstrom, David Koller, William Ribarsky, Larry Hodges, Nick Faust, and Gregory Turner. Real-time, continuous level of detail rendering of height fields. In Computer Graphics (SIGGRAPH '96), pages 109--118, August 1996.
4. Peter Lindstrom, David Koller, William Ribarsky, Larry Hodges, and Nickolas Faust. An integrated global gis and visual simulation system. 1997. submitted to Transactions on Visualization and Computer Graphics.
5. Ragunathan Rajkumar, Chen Lee, John Lehoczky, and Dan Siewiorek. A resource allocation model for qos management. 1997.
6. Vassilis J. Tsotras, Christian S. Jensen, and Richard T. Snodgrass. An extensible notation for spatiotemporal index queries. pages 47--53, 1998.
7. Gregory Turner, Jacques Haus, Gregory Newton, William Ribarsky, Larry Hodges, and Nick Faust. 4d symbology for sensing and simulations. In Proceedings of the SPIE Aerospace/Defense Sensing and Controls Symposium, pages 31--41.
8. Ouri Wolfson, Sam Chamberlain, Son Dao, Liqin Jiang, and Gisela Mendez. Moving objects databases: Issues and solutions . In the Proceedings of the tenth International Conference on Scientific and Statistical Database Management (SSDBM98), Capri (Italy), pages 111--122.
9. Ouri Wolfson, Sam Chamberlain, Son Dao, Liqin Jiang, and Gisela Mendez. Cost and imprecision in modeling the position of moving objects. In Proceedings of the 14-th International Conf. on Data Engineering, Orlando, FL., Feb. 1998.
10. Ouri Wolfson, A.Prasad Sistka, Son Dao, and Sam Chamberlain. Modeling and querying moving objects. In Proceedings of the 13-th International Conf. on Data Engineering, Birmingham, UK, April 1997.
11. K. Chakrabarti and S. Mehrotra, Efficient Concurrency Control im R-Trees, In Proceedings of the 14th international Conference on Data Engineering, Orlando, Fl, 1998.
12. K. Chakrabarti and S. Mehrotra, Hybrid Trees – a mechanism for indexing highly multidimensional databases, In Proceedings of the 15th International Conference in Data Engineering, 1999 (to appear).
13. K. Chakrabarti, S. Mehrotra, M. Ortega, J. Porkaew, and R. Winkler, A model for representing and processing uncertainty in databases. In Second Annual Symposium of the ARL federated Laboratory Display Consortium.
14. K. Porkaew and S. Mehrotra, Query Reformulation in MARS. (submitted for publication).